Porting COLA to AMD

EmPRiSM: a PaCER project

Waseem Kamleh

XXXIX International Symposium on Lattice Field Theory
Bonn, Germany, 8-13 August, 2022

Pawsey Supercomputing Centre

- Critical Australian infrastructure: Tier 1
- Critical infrastructure for Square Kilometre Array (SKA): Wajarri Yamatji country, 800km north of Perth
- AU$70m capital refresh by Australian Government
- Headquartered in Perth, Western Australia on Whadjuk country

CSIRO | Curtin University | Australia ECU Edith Cowan University | Murdoch University | The University of Western Australia | NCRIS National Research Infrastructure for Australia An Australian Government Initiative | Government of Western Australia

Pawsey Centre for Extreme Scale Readiness

Strategy

Achieving sustainable scalability on the new HPE Cray EX supercomputer will create a direct pathway to achieve superior scale on next-generation supercomputers

Collaboration/Co-investment

3-year partnerships with Pawsey and HPC vendors, early access to supercomputing tools and infrastructure, training and exclusive hackathons focused on HPC performance at scale.

Projects

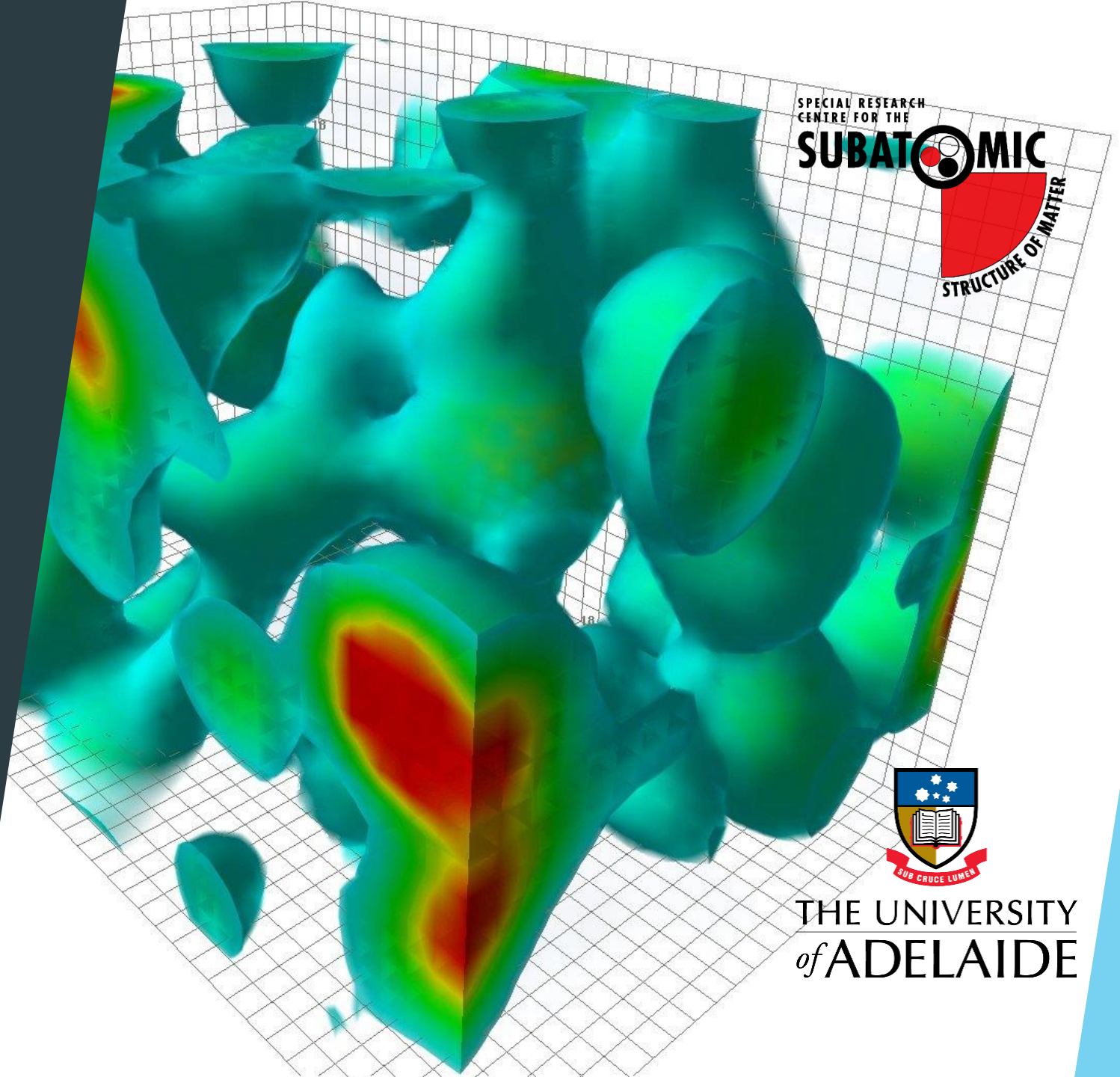
- 10 projects, 18 institutions
- Collaboration with US National Laboratories (ANL, ORNL, Ames) and industry (GE)

Emergent Phenomena Revealed In Subatomic Matter (EmPRiSM)

- How does the foundation of matter evolve to regimes relevant to the interior of neutron stars and the early universe?
- Can the centre-vortex structure of QCD ground-state fields account for the confinement of quarks and the dynamical generation of mass?
- How does the spectrum of hadron resonances emerge, and can a precision understanding guide the search for new physics?

Waseem Kamleh (UoA)
Derek Leinweber (UoA)
Ross Young (UoA)
James Zanotti (UoA)

Deva Deeptimahanti (Pawsey)



SPECIAL RESEARCH CENTRE FOR THE SUBATOMIC STRUCTURE OF MATTER

THE UNIVERSITY of ADELAIDE

Setonix HPE Cray EX

- 50 Pf next generation system
- Australia's most powerful research supercomputer
- Scientific Name for Quokka

Setonix's Computational Capability

- Significant increase in computational capacity
- Increase in double precision floating point operations available are:

Magnus

1.1 petaflops

Setonix phase 1

2.7 petaflops

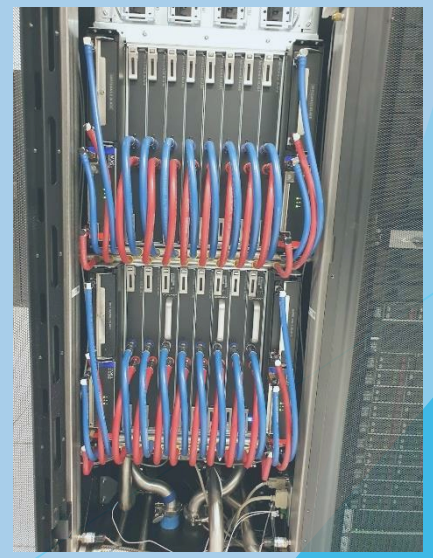
Setonix phase 2

50+ petaflops

Setonix

HPE Cray EX

CPU – Compute Nodes	3200 AMD Milan, 64 cores, 2 CPU/Node
CPU – GPU Nodes	192 AMD 64 cores, InfinityFabric, 1 CPU/Node
GPU	**768 AMD NextGen Instinct**, 128 GB, InfinityFabric, 4 GPU/Node
Interconnect - Compute Nodes	200Gb/s - 400 Gb/s
Interconnect – GPU Nodes	800 Gb/s
Scratch Filesystem	ClusterStor E1000: Lustre, 15 PB (~3.5 PB NVMe, 11.5 PB HDD), 880GB/s Flash, 120 GB/s HDD
Near Node Filesystem	ClusterStor E1000: Lustre, ~1PB, 1.4 TB/s
Estimated Peak Power	1.93 MW
Estimated Typical Power	1.63 MW
System Management	Shasta: RedFish Interface, Microservices, Containers, Orchestrator.



SHATTERING PERFORMANCE BARRIERS IN HPC & AI

PEAK PERFORMANCE	A100	MI200*	INSTINCT™ ADVANTAGE
FP64 VECTOR	9.7 TF	47.9 TF	**4.9X**
FP32 VECTOR	19.5 TF	47.9 TF	**2.5X**
FP64 MATRIX	19.5 TF	95.7 TF	**4.9X**
FP32 MATRIX	N/A	95.7 TF	N/A
FP16, BF16 MATRIX	312 TF	383 TF	**1.2X**
MEMORY SIZE	80 GB	128 GB	**1.6X**
MEMORY BANDWIDTH	2.0 TB/s	3.2 TB/s	**1.6X**

NOTE: THE A100 TF32 DATA FORMAT IS NOT IEEE FP32 COMPLIANT , SO NOT INCLUDED IN THIS COMPARISON.

AMD MI200*
November 2021

47.9 Teraflops (DP)

0.5 kW

58 Billion Transistors

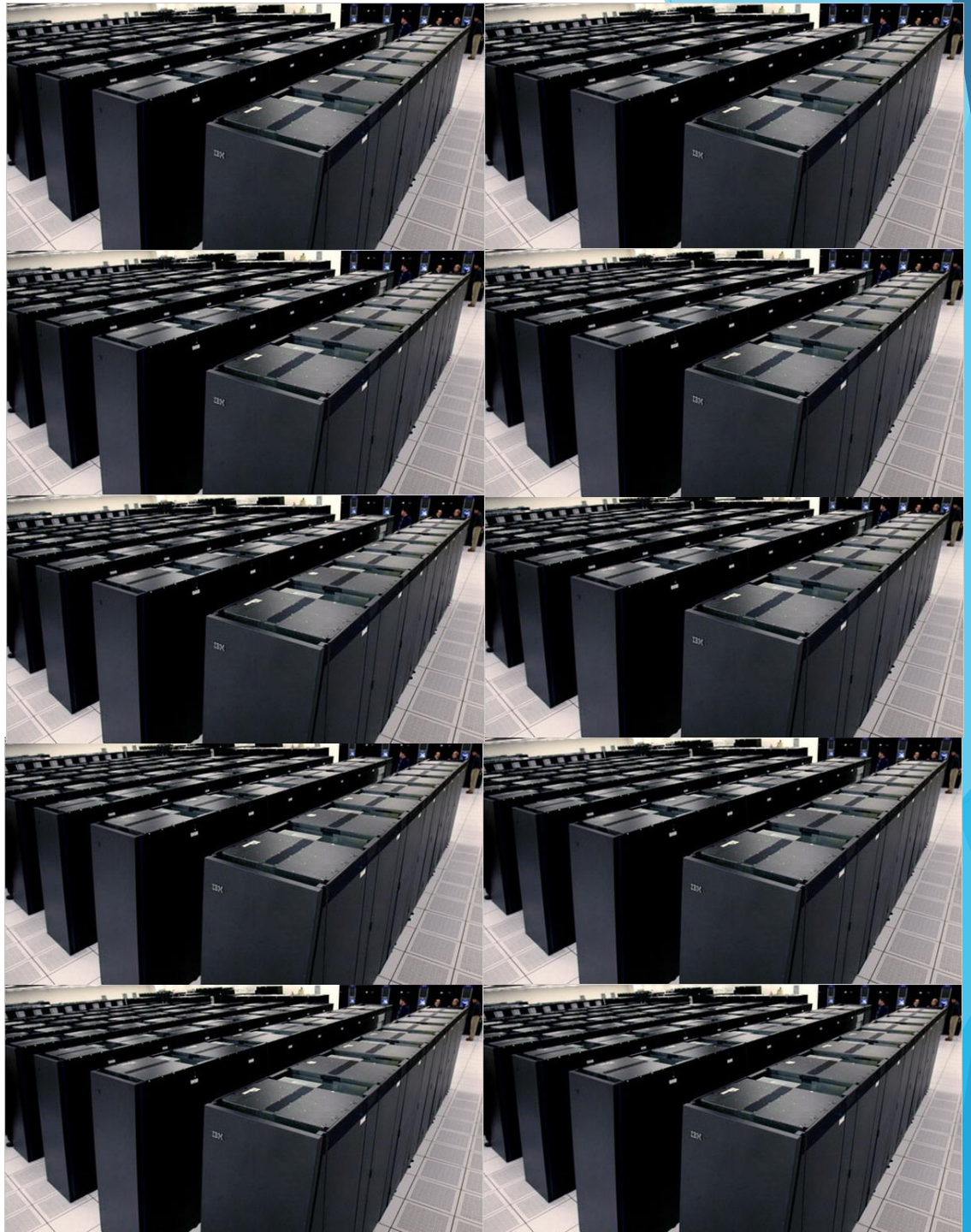
IBM ASCI WHITE
#1 November 2000

4.9 Teraflops (DP)

3000 kW

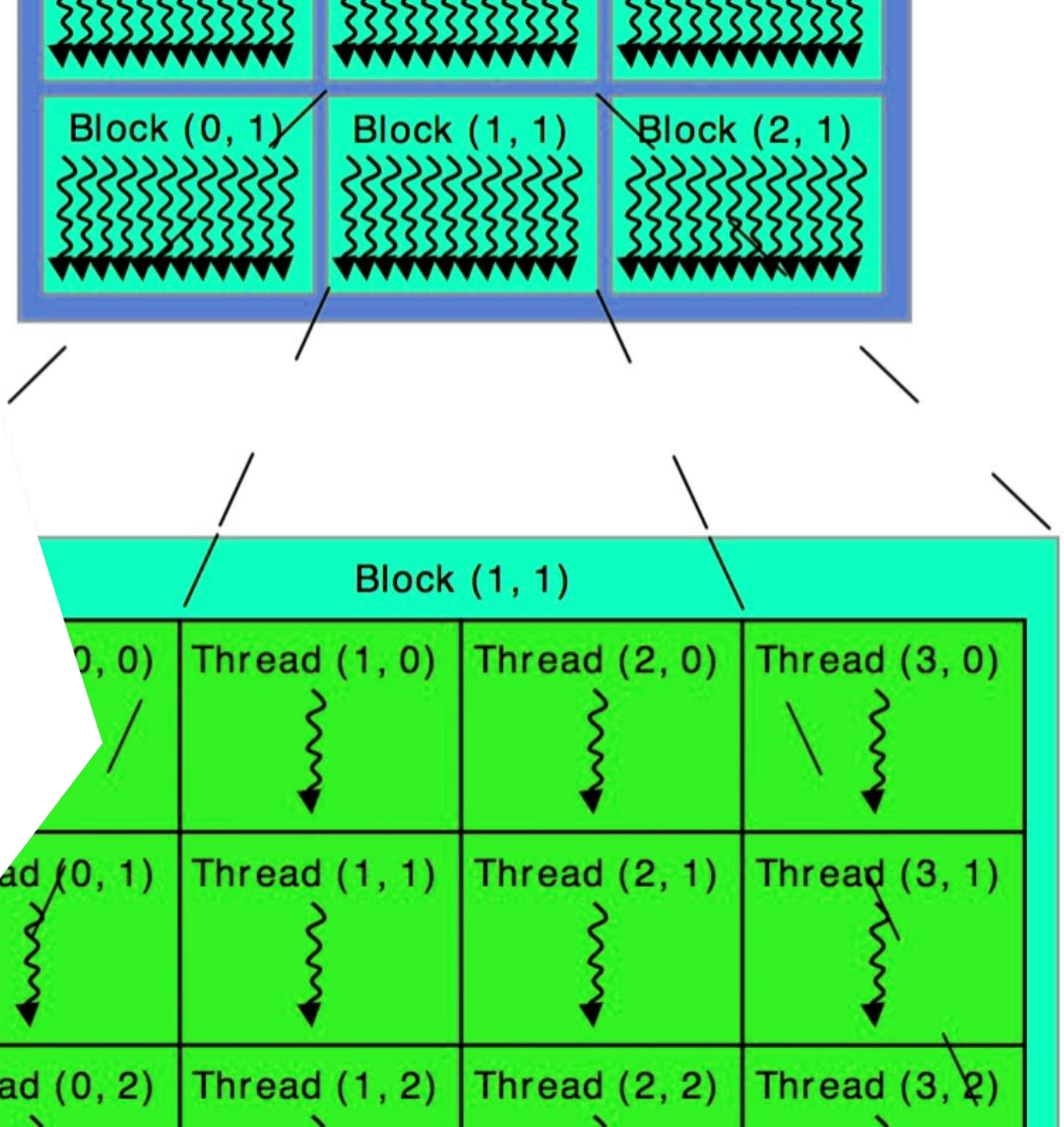
188 Billion Transistors


AMD
INSTINCT



COLA

- Custom in-house code
  - Started development ~20 years ago
  - Low flops/byte=bandwidth limited
  - Memory footprint is relatively small
- Modern Fortran (>2008) + MPI
  - + CUDA C/C++ for GPU acceleration
  - Mixed language programming
  - Inverter, HMC, eigenmodes
- Intrinsic geometric parallelism
  - Map sublattice to a CPU core
  - Map one site to a GPU thread
  - 20X speedup on GPUs



```
template double vfgpu::normsq<dcomplex>(const dcomplex* left);

template<typename _complex_a,typename _complex_b>
double vfgpu::real_inner_product(const _complex_a* left, const _complex_b* right)
{
  size_t Ns_part = tPB_x*sizeof(double);
  size_t Ns_sum =  tPB_sum *sizeof(double);
  double inner;

  PartialRealInnerProduct<<<bPG_x,tPB_x,Ns_part>>>(left,right,nb_vf);
  TreeReduceBlockSumReal<<<1,tPB_sum,Ns_sum>>>(nb_sum);
  ErrorCheck(cudaMemcpyFromSymbol(&inner, resultReal, sizeof(double)));

  return (inner);
}

template double vfgpu::real_inner_product<complex,complex>(const complex* left, const complex* right);
//template double vfgpu::real_inner_product<complex,dcomplex>(const complex* left, const dcomplex* right);
//template double vfgpu::real_inner_product<dcomplex,complex>(const dcomplex* left, const complex* right);
template double vfgpu::real_inner_product<dcomplex,dcomplex>(const dcomplex* left, const dcomplex* right);

template<typename _complex_a,typename _complex_b>
dcomplex vfgpu::inner_product(const _complex_a* left, const _complex_b* right)
{
  size_t Ns_part = 2*tPB_x*sizeof(double);
  size_t Ns_sum =  2*tPB_sum *sizeof(double);
  dcomplex inner;

  PartialInnerProduct<<<bPG_x,tPB_x,Ns_part>>>(left,right,nb_vf);
  TreeReduceBlockSumComplex<<<1,tPB_sum,Ns_sum>>>(nb_sum);
  ErrorCheck(cudaMemcpyFromSymbol(&inner, resultComplex, sizeof(double2)));

  return (inner);
}

template dcomplex vfgpu::inner_product<complex,complex>(const complex* left, const complex* right);
//template dcomplex vfgpu::inner_product<complex,dcomplex>(const complex* left, const dcomplex* right);
//template dcomplex vfgpu::inner_product<dcomplex,complex>(const dcomplex* left, const complex* right);
template dcomplex vfgpu::inner_product<dcomplex,dcomplex>(const dcomplex* left, const dcomplex* right);

template<typename _complex_a,typename _complex_b>
-=--:----F1  vectorfields_gpu.cu   25% L430  SVN@0  (C++/l Abbrev) ----------
```

Porting to AMD architecture

- CPU code written in Fortran 2008
  - Just needs a compiler
- Optimisations
  - Vector instructions
  - Slingshot interconnect
- GPU routines written in CUDA
  - Initially HIPify to run on AMD
  - Explore other language options
  - Portability vs performance

Technical goals

Cross-platform

- In-house code
- Optimise for multiple architectures
- Ideally maintain a single source tree

CPU code

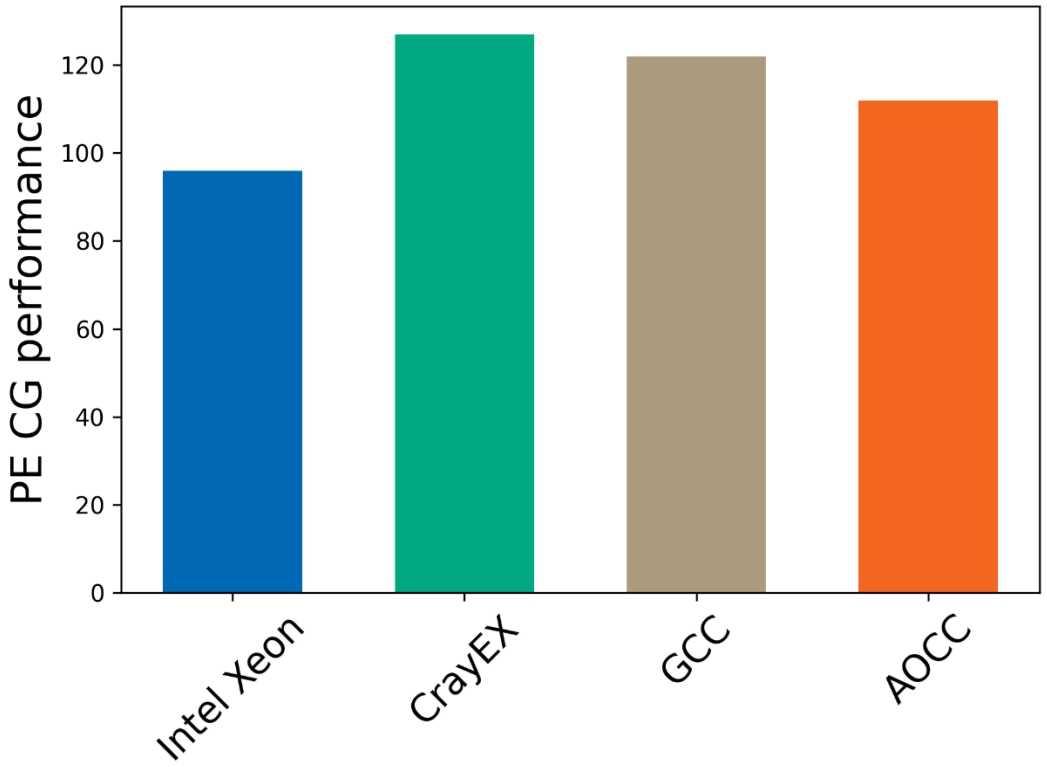
- Fortran '08 (platform independent)
- Optimise per-core performance
- Optimise communication patterns

GPU acceleration

- Convert Nvidia CUDA to an AMD API
- Optimise kernel performance
- Multi-GPU over Infinity fabric

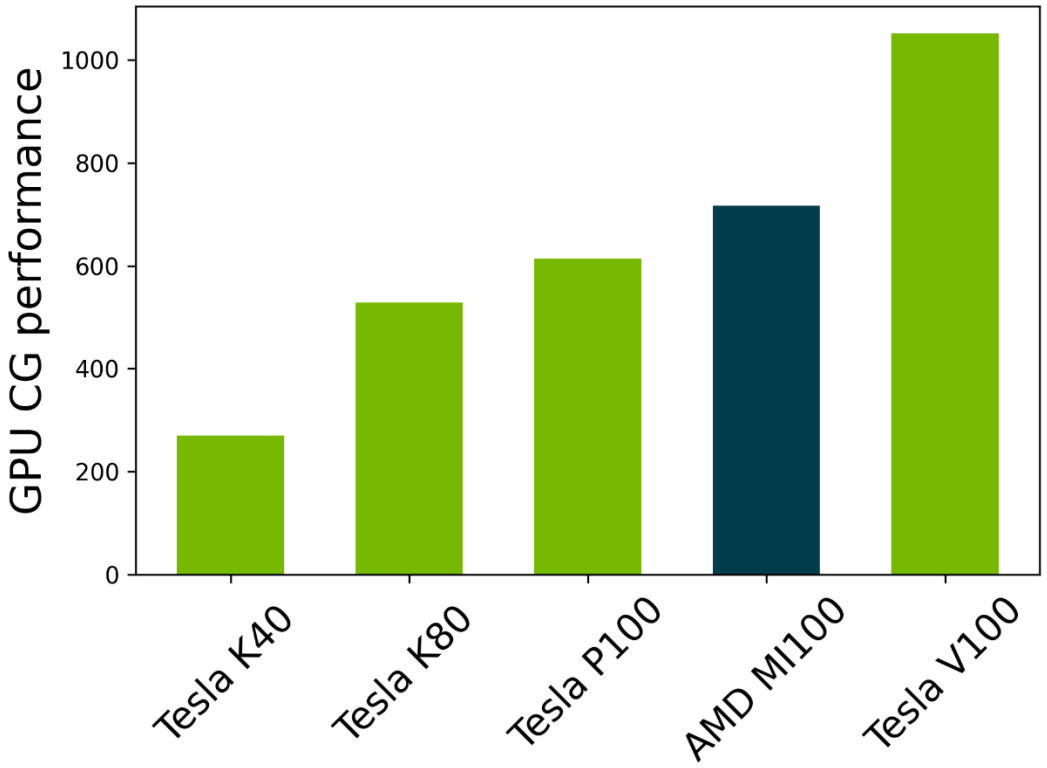
Programming Environment

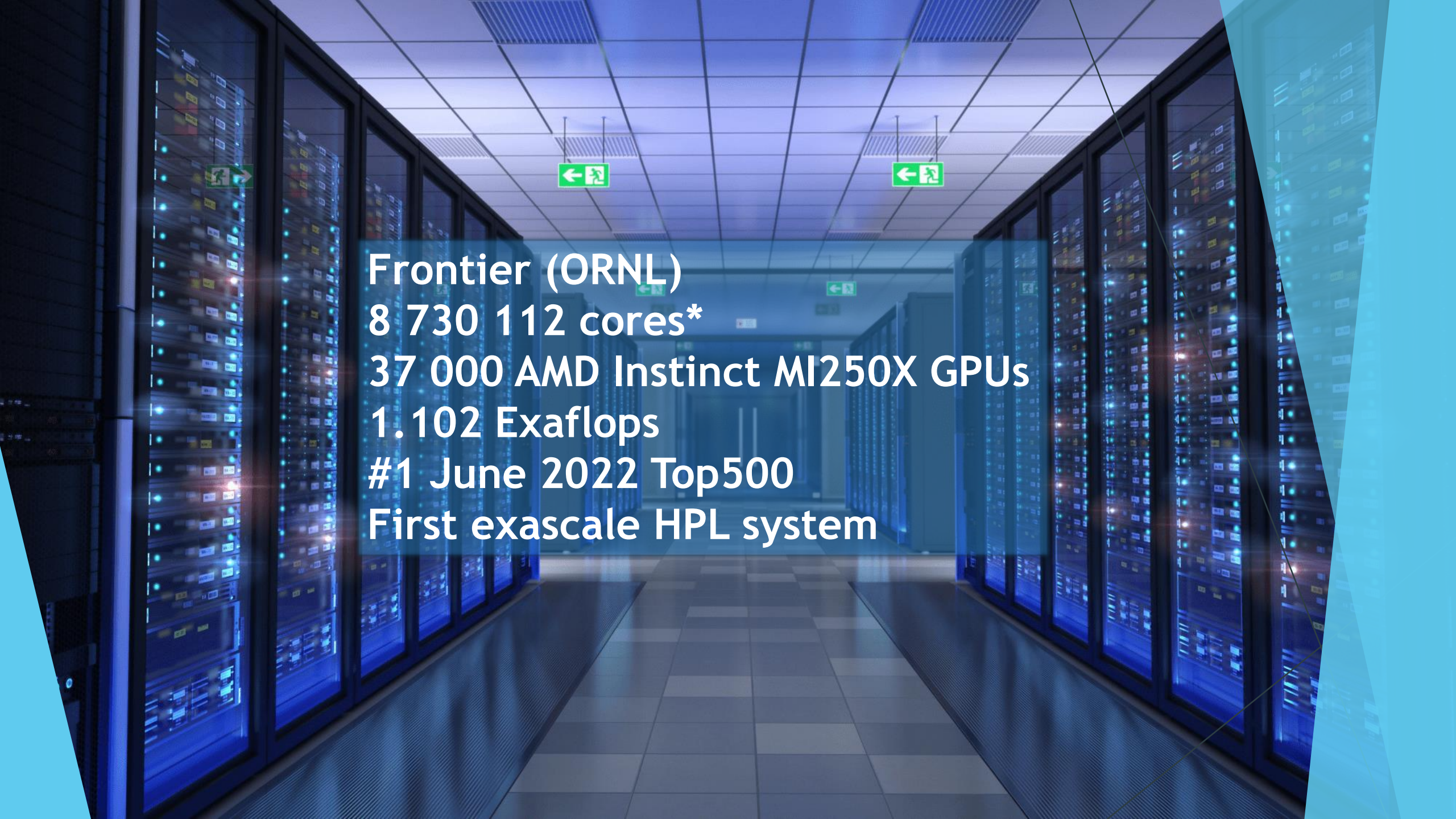
- Intel PE/ifort
  - “Porting” from here
  - Cascade Lake Xeon (48 core) baseline
- CrayEX/ftn [best]
  - Fortran pre-processor != cpp
  - Stricter standard enforcement
- GCC/gfortran [good]
  - Minor tweaks to logical semantics
- AOCC/flang [avoid]
  - Experimental/buggy
  - Poor modern Fortran support



HIPification

- Porting from CUDA platform
- HIPified with minor changes
- Same performance using CUDA and HIP on NVIDIA V100 nodes
- Platform portability
- Performance portability?





Frontier (ORNL)
8 730 112 cores*
37 000 AMD Instinct MI250X GPUs
1.102 Exaflops
#1 June 2022 Top500
First exascale HPL system

Summary

- Focus on CPU code optimisations until AMD MI250X* GPUs arrive as part of phase 2b in 2023.
- Use meta-programming in Python to explore different data layouts and code factoring.
- PaCER presents an exciting opportunity to conduct extreme scale simulations on Setonix.
- Harnessing the power of GPU acceleration is critical to address the project computational challenges.